Introduction to Graphics

Assignment 3: Projections

Student: Kasper Passov

1 Problem Description

Projections is a group of protocala drawing objects from a 3d world onto a 2d plane. There are two catagories of projection, each with its own protocols:

- Parallel projection: line of sight is a plane parallel to the projection plane.
- Perspective projection: line of sight originates from a point.

This gives perspective projection the interesting feature of gaining a parllel projection if the distance between the camera and the projection plane is raised into infinity.
The given assignment has us implementing a method of transforming an arbitrary view volume, into a canonical parallel view volume. This gives us two types of transformations, one from an arbitrary parallel view, and one from an arbitrary perspective view.

2 Implementaion Theory

2.1 Parallel projection

A transformation from a parallel projection into a canonical parallel view has 4 steps:

- Translate VRP(View reference point) to the origin
- Rotate the *Eye-Coordinate System*
- Shear so the DOP(Direction of projection) is parallel to the Z-axis
- Translate and scale to the *Canonical Orthographic View Volume*

Step 1: Translate VRP to the origin Is pretty simply a translation with the negative VRP vector, as the VRP vector takes base in origin.

$$T(-VRP) = \begin{bmatrix} 1 & 0 & 0 & -vrp_x \\ 0 & 1 & 0 & -vrp_y \\ 0 & 0 & 1 & -vrp_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 2: Rotate the *Eye-Coordinate System* We want our (u, v, n) axis to line up with the (x, y, z) axis, this is done by the rotation based on VPN. This is done by the matrix:

$$R = \begin{bmatrix} r_{1x} & r_{2x} & r_{3x} & 0 \\ r_{1y} & r_{2y} & r_{3y} & 0 \\ r_{1z} & r_{2z} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where our R vectors are defined as:

$$R_z^T = (r_{1z}, r_{2z}, r_{3z}) = \frac{VPN}{||VPN||_2}$$
$$R_x^T = (r_{1x}, r_{2x}, r_{3x}) = \frac{VUP * R_z}{||VUP * R_z||_2}$$
$$R_y^T = (r_{1y}, r_{2y}, r_{3y}) = \frac{R_z * R_x}{||R_z * R_x||_2}$$

This gives us a projection based in the origin of (x, y, z) coinsiding with the axises.

Step 3: Shear the DOP We now wish to shear the matrix so our DOP is parallel to our z-axis. The DOP is defined as the length between our View Reference Point and the Center Window.

$$DOP = PRP - CW = (dop_u, dop_v, dop_n, 0)^T$$

To do the transformation we use the following transformed shear matrix:

$$Sh_{par} = \begin{bmatrix} 1 & 0 & -\frac{dop_u}{dop_n} & 0 \\ 0 & 1 & -\frac{dop_v}{dop_n} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This gives us a projection based in the origin and coinsiding with the axes sof (x, y, z), with a DOP parallel with the z-axis.

Step 4: Translate and scale The last part of our parallel projection transformation, is a translation and scalation into the *Canonical Parallel View Volume* Moving the front of our view plane onto the XY-plane, and placing CW at the origin. CW is by definition in the center of our view plane (u, v):

$$CW = (cw_u, cw_v, 0)^T = (\frac{u_{max} + u_{min}}{2}, \frac{v_{max} + v_{min}}{2}, 0)$$

We translate into our CW

$$T_{par} \begin{bmatrix} 1 & 0 & 0 & \frac{u_{max}+u_{min}}{2} \\ 0 & 1 & 0 & \frac{v_{max}+v_{min}}{2} \\ 0 & 0 & 1 & -F \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Lastly we scale our view volume into the dimensions

$$[-1, 1] \cdot [-1, 1] \cdot [0, -1]$$

With the matrix

$$S_{par} \begin{bmatrix} \frac{2}{u_{max}+u_{min}} & 0 & 0 & 0 \\ 0 & \frac{2}{v_{max}+v_{min}} & 0 & 0 \\ 0 & 0 & \frac{1}{F-B} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.2 Perspective projection

A transformation from a perspective projection into a canonical parallel view has 5 steps:

- Translate VRP to the origin
- Rotate the *Eye-Coordinate System*
- Translate PRP to the origin
- Shear so the center line of the *View Volume* is parallel to the Z-axis
- Scale to the *Canonical Perspective View Volume*

This has some in common with the parallel projection, as step one, two and 4 are calculatet the same way, so these will be skipped in this explanation.

Step 3: Translate PRP to origin We translate our PRP using the same method as in step 1, but with (u, v, n) as our perspective is based on that coordinate system.

$$T(-PRP) = \begin{bmatrix} 1 & 0 & 0 & -prp_u \\ 0 & 1 & 0 & -prp_v \\ 0 & 0 & 1 & -prp_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 4: Shear center line of the *View Volume* to the Z-axis We wish to move the center of the View Volume to the Z-axis, this will be the same transformation as in parallel projection as the DOP vector has the same slopes as our perspective center line.

Step 5: Scale to the *Canonical Perspective View Volume* The last step involes scaling into the Canonical Perspective View Volume and is done by the matrix

$$S_{par} = \begin{bmatrix} \frac{-2prp_n}{(u_{max}-u_{min})(B-prp_n)} & 0 & 0 & 0 \\ 0 & \frac{-2prp_n}{(v_{max}-v_{min})(B-prp_n)} & 0 & 0 \\ 0 & 0 & \frac{-1}{B-prp_n} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3 Summation

Parallel The above transformation can be summed up as:

$$N_{par} = S_{par} \cdot T_{par} \cdot Sh_{par} \cdot R \cdot T(-VRP)$$

Creating a *Canonical Parallel View Volume* from a *World Coordinate System.*

Just for fun we left-multiply this by the following matrix to get the *Orthographic Projection*

$$\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

and lastly leftmultiply by the *Window-Viewport* matrix:

$$M_{WV} = S(\frac{width}{2}, \frac{height}{2}, 1) \cdot T(1, 1, 0)$$

This last matrix translates and scales our **Canonical Window** into a **Actual Screen View Port**. Giving us the final transformations:

$$M_{par-total} = M_{WV} \cdot S_{par} \cdot T_{par} \cdot Sh_{par} \cdot R \cdot T(-VRP)$$

Perspective The above transformations can be summed up as, transforming a *World Coordinate System* to a *Canonical Perspective View Volume*:

$$N_{par} = S_{per} \cdot Sh_{per} \cdot T_{-PRP} \cdot R \cdot T(-VRP)$$

Just for fun we wish to transform our *Canonical Perspective View Volume* to the *Canonical Orthographic Projection* using the matrix:

$$M_{perpar} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1+Z_{max}} & \frac{-Z_{max}}{1+Z_{max}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

and lastly leftmultiply by the *Window-Viewport* matrix:

$$M_{WV} = S(\frac{width}{2}, \frac{height}{2}, 1) \cdot T(1, 1, 0)$$

This last matrix translates and scales our **Canonical Window** into a **Actual Screen View Port**. Giving us the final transformations for *World-Coordinate* to *Actual Screen View Port*:

$$M_{par-total} = M_{WV} \cdot M_{perpar} \cdot S_{per} \cdot Sh_{per} \cdot T(-PRP) \cdot R \cdot T(-VRP)$$

3 Implementaion

To draw the house i implemented each of the three vertices in the GLTriangle doing the following for each, with the exact reason commented in the code-examples.

```
void GLTriangle::draw() {
    ...
    glBindBuffer(GL_ARRAY_BUFFER, m_BackVBO); // Binds the VBO to the buffer
    glVertexAttribPointer(m_vertexPositionAttribute, 3,
                          GL_FLOAT, GL_FALSE, 0, 0);
    glDrawArrays(GL_LINE_LOOP, 0, m_numberOfVerticesBack); //draws
    ...
    }

void GLTriangle::initializeBuffers(ShaderProgram & shaderProgram) {
    ...
    m_numberOfVerticesFront = 5; //number of vectors

    float front[] = { //the positions of all points
        0.0f, 0.0f, 54.0f,
        16.0f, 0.0f, 54.0f,
        16.0f, 10.0f, 54.0f,
        8.0f, 16.0f, 54.0f,
        0.0f, 10.0f, 54.0f
    };

    glGenBuffers(1, &m_FrontVBO); //generates buffer
    glBindBuffer(GL_ARRAY_BUFFER, m_FrontVBO); //binds the buffer
    glBufferData(GL_ARRAY_BUFFER, sizeof(front),
                 front, GL_STATIC_DRAW); // binds buffer data
    glVertexAttribPointer(m_vertexPositionAttribute, 3,
                 GL_FLOAT, GL_FALSE, 0, 0);
    ...
    }
```

This with the translation and perspective

```
modelMatrix = glm::translate(modelMatrix, glm::vec3(0.0f, 0.0f, -100.0f));
glm::mat4 perspectiveMatrix = glm::perspective(45.f, 1.f, 2.0f, 120.f);
```

draws the following house.